CLAMP: Active queue management at wireless access points [1]

Lachlan L. H. Andrew, Stephen V. Hanly and Rami G. Mukhtar

ARC Special Research Centre on Ultra-Broadband Information Networks
(an affilliated programme of the National ICT Australia),
Department of Electrical and Electronic Engineering, University of Melbourne, Australia
e-mail: {lha,hanly}@ee.mu.oz.au

***Abstract*: This paper investigates the performance of CLAMP, a distributed algorithm to enhance the performance of TCP connections that terminate in a wireless access network. CLAMP works at a receiver to control a TCP sender by setting the TCP receiver's advertised window limit. CLAMP provides explicit control over wireless link utilisation and queueing delay at the access point buffer, and can drastically increase the throughput of so-called short TCP flows with negligible loss in long TCP flow throughput.**

1. Introduction

TCP Reno, and its variants, are the dominant transport layer protocols for data transfers in the internet. It has been a remarkably successful protocol; its simple and scalable bandwidth probing, coupled with reliable packet delivery, have provided the robustness required for rapid growth. Nevertheless, it has long been known that it has some weaknesses, such as unfairness between flows with disparate propagation delays. More recently, attention has been drawn to the throughput performance of short flows, or "mice", as compared to long lasting "elephant" flows, where it has been observed that the mice suffer from the presence of elephants, especially when network buffers are large. This is a significant issue, as most TCP flows are actually small data transfers, such as those involved in web browsing. Further, other real-time flows may share the same network buffers as TCP flows, where latency is even more critical. In the present paper, we investigate a new technique to overcome the deficiencies of the current protocol, without necessitating the complete replacement of the protocol with a new one.

In the present paper, we focus on the performance of wireless networks, where mobile terminals are the receivers, downloading information from servers located anywhere in the Internet via a common access point (AP). We take the point of view of a wireless network provider, who has no control over the particular versions of TCP that are installed on the servers in the Internet, but who has control of the wireless technology to be used in the access network, including the AP, and the wireless terminals. We use the receiver-side approach, and provide a control algorithm that allows the AP to control the amount of queueing that occurs in the AP buffers. The advantages of this approach are that the tradeoff between mice and elephant throughput can be tuned to the particular characteristics of the wireless network, and that weighted fairness between the elephants can be enforced.

Most work on TCP over wireless channels has focussed on the issue of packet loss. It is well known that TCP Reno treats packet loss as a sign of network congestion, and reacts by halving the congestion window whenever packet loss is detected. This is unfortunate if the packet loss is due to wireless channel conditions being temporarily unfavourable, rather than due to genuine congestion. There are now many solutions available to overcome this problem.

Mechanisms exist at layers 1 and 2 to handle channel impairments that would otherwise cause wireless packet loss. These include coding and signal processing methods, and layer 2 mechanisms such as link rate adaptation ([1]), and incremental redundancy ([2] and references therein), which adapt the coding rate and modulation to changing channel conditions. Link layer retransmissions can hide wireless-related losses from the end hosts by locally retransmitting packets lost across the wireless channel [3–8]. The net effect of these mechanisms is to provide a low packet loss rate at the expense of variable packet transmission times [5, 9]. In the present paper, we assume a solution to the wireless packet loss problem is available, and mainly consider IP packet loss rates that are small (such as 10^{-3}).

The above lower-layer optimizations change the focus of TCP research from that of mitigating the effect of errors, to mitigating the effect of queueing delay, and this is the main purpose of CLAMP. One particular problem with variable packet delay and queueing variation is an increased incidence of TCP time-outs, with a significant negative impact on TCP throughput. This provides a very strong case for trying to control the queueing delay at the AP.

The algorithm, CLAMP, that we investigate in this paper, is shown to be very beneficial as a flow control mechanism. We consider its operation at the receiver side, in conjunction with TCP Reno at the sender-side, and compare its performance against that of TCP Reno without CLAMP. We show that CLAMP can improve the throughput of long-lasting flows, by preventing buffer overflow at the AP, and allowing lower layer optimizations to take effect. It can be configured to provide fairness between long-lasting flows, irrespective of

[1] This work was supported by the Australian Research Council.

their propagation delays, and it can provide a drastic improvement in the throughput of mice flows, which include small TCP flows who spend most of the time in slow start. This is achieved because CLAMP is able to control the latency at the AP.

2. System Topology and Design Constraints

2.1. Design Constraints

The scheme we investigate, CLAMP, has been proposed in [10], but the performance analysis of this algorithm to date has focussed on that of TCP flows over low bandwidth channels, that are slowly time-varying in capacity, but otherwise error free. The present paper provides new results for wireless channels, using a simulation model that includes the phenomena of wireless fading, link layer retransmissions, and IP packet loss. This study enables us to understand the impact of Doppler effects, and the impact of link layer retransmissions. Furthermore, in this paper we demonstrate the tradeoff between small and long TCP flow throughput, which provides a startling illustration of the benefits of the controlled queueing that CLAMP provides. We begin with a review of the basic network model and provide a precise statement of the CLAMP protocol [10].

2.2. Topology and Notation

The access network topology of interest is illustrated in Figure 1. For the sake of illustration we have depicted the access link as a wireless network access point; however, it can be any one of a variety of wireless technologies as discussed in the introduction. In the example scenarios studied in this paper, the access point maintains a separate queue for each user, to allow channel-state-aware scheduling between users, but users may be involved in multiple TCP sessions which then share the same queue, as is the case for user 1 in Figure 1. Alternatively, the access point may use only a single queue for all users (not depicted, but relevant for wireline applications).

The service rate of a queue, μ_c bytes/sec, depends on the channel statistics of the users, and the scheduling policy at the wireless access point. Referring to Figure 1, each flow, i, has a source node, S_i, and an associated round trip transmission delay, which includes all propagation and queueing delays, except for the queueing delay at the access router.

Each sending node implements sliding window flow control. Under the assumption that sources are greedy, the total number of packets and acknowledgements for flow i, in flight at any time, t, is equal to the window size, $w_i(t)$. The CLAMP algorithm selects $w_i(t)$ in a decentralized way, such that each flow sharing the same queue obtains a proportional share of the service rate, μ_c, and the equilibrium buffer occupancy of the queue, $q(t)$, can be controlled as described below.

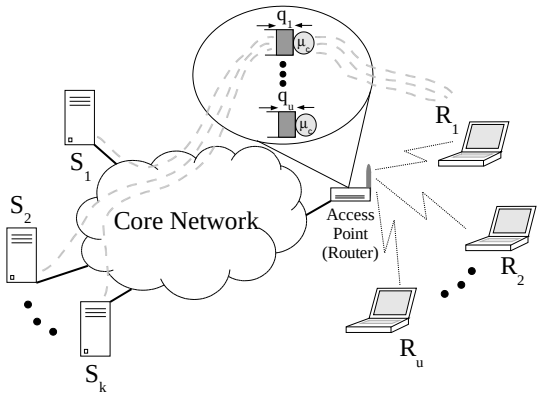


Figure 1: System topology.

3. The CLAMP Protocol

In order to implement CLAMP, the access network should be modified by inserting a software agent in both the access router and the client, as described below. As an aside, note that it is also possible to implement CLAMP as a performance enhancing proxy (PEP), with the associated inability to use IPsec; indeed, the precursor to CLAMP described in [11] has been implemented as a GPRS PEP, with very promising results [12].

3.1. Access Router Agent

The software agent in the access router periodically samples each user's queue length, and computes a moving average, q. Given q, it evaluates a measure of "congestion", $p(q)$: the monotonic increasing function of q given below. The value is then passed to each receiver, either by inserting the value into the IP header of each packet leaving the access router, or by each receiver explicitly requesting the value from the access router as required. Note that using the IP header rather than the TCP header allows IP security to remain unaffected. The excessive routing delay incurred by packets containing IP options will not be a problem, since the option is only present in the few hops between the access router and the receiver; here routing is mostly done in software, and so options do not interfere with a hardware switching fabric.

This paper uses the piecewise linear function

$$p_s(q) = \max(0, b(q-a)/\mu_c), \qquad (1)$$

although it is possible to use other increasing functions with bounded slope. The dimensionless constant $b < \pi/2$ determines how sensitive the bottleneck queue size is to the number of flows (see [13]). The parameters a (in bytes) and b control the equilibrium mean queue size, q^*, as shown in [10]. They can be statically set or dynamically tuned to obtain a desired queueing behavior.

3.2. Client Side Agent

The client agent's function is to receive the router's congestion measure, p, and set the receiver's TCP AWND value according to the algorithm described below, hence clamping the sender's TCP transmission window.

If IP security is not required, this agent can be implemented as a wrapper for the software driver for the wireless network interface by modifying packets' payloads. Alternatively, using an open-source protocol stack, the price information can be passed up from the IP layer to the TCP layer, and the agent can operate securely at the transport layer.

3.3. Window Update Algorithm

For simplicity, the algorithm will be described for a single flow. Let t_k denote the time instant when the kth packet, of size s_k bytes, is received by the receiving client. CLAMP calculates the change in window size (in bytes [14]) as

$$\Delta w(t_k) = \left[\frac{\phi\tau - p(q(t_k))\hat{\mu}(t_k)}{\hat{d}(t_k)} \right] (t_k - t_{k-1}) \quad (2)$$

where $\hat{\mu}$ is an estimate of the average received rate of the flow in bytes per second, $\hat{d}(t_k)$ is given below, $\phi > 0$ is a positive constant expressing the flow's need for bandwidth, and $\tau > 0$ (bytes) is a constant for all flows. The term in τ tries to increase the window at a constant rate, to balance the term in $\hat{\mu}$ which reduces it at a rate which increases with the occupancy of the queue and with the proportion of traffic due to the particular flow.

The current received rate, $\hat{\mu}$, is estimated using a sliding window averaging function,

$$\hat{\mu}(t_k) = \frac{\alpha \sum_{i=k-\alpha}^{k} s_i}{t_k - t_{k-\alpha}}, \quad (3)$$

and

$$\hat{d}(t_k) = (1-\beta)\hat{d}(t_{k-1}) + \beta \frac{\text{AWND}(t_k)}{\hat{\mu}(t_k)}, \quad (4)$$

where the integer α and $\beta \in (0,1)$ are smoothing factors, and AWND(t_k) is the actual advertised window (see below). These estimators were chosen for their simplicity, and other estimators may prove to be more effective. In equilibrium, $\hat{d}$ will be the RTT of the flow in seconds, i.e., propagation plus queueing delay. However, computing $\hat{d}(t_k)$ does not require an explicit measurement of such delay, and the algorithm achieves its objectives even if it is not an accurate estimate of the RTT.

Note that for notational ease we have omitted the index of the flow in the above notation. However, each flow will maintain its own $\phi, w, \hat{\mu}$, and $\hat{d}$.

Before updating the window size, w, Δw is clipped, giving

$$w(t_k) = \begin{cases} w(t_{k-1}) - s_k & \text{if } \Delta w(t_k) < -s_k \\ w(t_{k-1}) + \Delta w(t_k) & \text{if } -s_k \le \Delta w(t_k) \le \bar{\Delta} \\ w(t_{k-1}) + \bar{\Delta} & \text{if } \Delta w(t_k) > \bar{\Delta} \end{cases} \quad (5)$$

The maximum window increase, $\bar{\Delta} > 0$, prevents large $t_k - t_{k-1}$ from causing large changes in w when packets arrive infrequently, such as when a source becomes idle for an extended period of time. Limiting the decrease in window size to the received packet size ensures the right edge of the window is monotonic non-decreasing [14, p. 42].

The AWND value advertised to the sender is then

$$\text{AWND}(t_k) \equiv \min(w(t_k), \text{AWND}), \quad (6)$$

where the AWND on the right hand side of the assignment is the value received by the client side agent from the receiver's operating system. The assigned value on the left hand side is the one to be used in (4).

The flow control algorithm can provide non-uniform sharing of the bottleneck bandwidth by appropriately setting the constants ϕ_i. In [10], it is shown that, under certain conditions, flow i will obtain the proportion $\phi_i / \sum_{j=1}^{k} \phi_j$ of the bottleneck capacity, where k is the number of flows sharing the bottleneck. In the present paper, we shall set all $\phi_i = 1$, to obtain a fair allocation of capacity amongst competing flows.

3.4. Slow Start

It is important for CLAMP to allow TCP slow-start to take place whenever necessary (at the start of a connection or after a time-out) to fill the network pipe. For this reason, we include a receiver-side slow start to mirror the sender-side behaviour. Due to lack of space, we do not describe this mechanism in detail here, suffice to say that we terminate the receiver-side slow-start at packet loss (detected by three duplicate acknowledgements) or when $\Delta w < 0$, whichever occurs first.

4. Performance Evaluation

This section demonstrates by simulation that CLAMP improves the throughput of short flows and provides fairness for flows with differing round trip times. In the context of wireline networks, it has already been shown [10] that CLAMP relinquishes control correctly to TCP when there is congestion elsewhere in the network.

4.1. Wireless channel model

CLAMP aims to ensure that fluctuations in a wireless link do not adversely affect the higher layers. The model used in these simulations aims to be as simple as possible, while describing the variable rate transmission characteristic of optimal wireless links. Some simulations model the impact of packet loss. Others assume either that link layer retransmission results in a lossless link,

or that a scheme such as TCP Veno is being used, which disregards wireless loss and responds only to buffer overflow.

The path loss fading process between the base station and each receiver is a stationary, stochastic process with marginal statistics given by the Rayleigh distribution. Jakes' model is used with $G = 8$ generators and a Doppler frequency of $f_d = 0.6$ Hz (1.8 GHz carrier, 0.1 m/s mobile speed). That gives a channel of square magnitude

$$N(s) = \left| \sqrt{\frac{2}{G}} \sum_{i=1}^{G} \cos(\omega_i t) \exp\left[\frac{j\pi i}{G}\right] \right|^2,$$

where $\omega_i = 2\pi f_d \cos(2\pi(i + 0.5)/(4G))$.

Frame transmissions over the physical channel occur in slots of $\tau_{\text{slot}} = 3$ ms; a slot is the time period of one frame transmission attempt. The process is sampled on a slot by slot basis. Rate matching is used, and in slot s, $B(s)$ information bits are transmitted:

$$B(s) = \tau_{\text{slot}} \min\left((1 - m)W_0 \log(1 + N(s)), C_{\max}\right),$$

where $C_{\max}$ is the maximum possible rate that can be transmitted over the channel, $N(s)$ is the SNR at the start of the frame, W_0 models the bandwidth, and m is a "backoff margin" described below.

In order to capture the bursty nature of errors we use a simple error model: if the SNR, $N(e)$, at the end of the frame is such that

$$B(s) < \tau_{\text{slot}} W_0 \log(1 + N(e)),$$

then an error is declared. This will occur if the channel deteriorates sufficiently during the slot that it can no longer support the selected rate. If the SNR is sufficiently high at both the start and end of a slot, then it is assumed to be sufficiently high throughout the slot. The backoff margin, m, is included to allow some degradation to occur without causing an error.

If a frame is received in error, then the base station calculates a new rate based on the new SNR. It retransmits as much of the frame as can be accommodated in one slot at the new rate, with the remaining bits being queued to form part of the next frame. A maximum of $R = 3$ slot transmissions are allowed. If after R attempts the frame is still received in error, then the erroneous frame is delivered as is to higher layers, ultimately resulting in a packet error. The backoff parameter is adaptively tuned such that the packet error rate after retransmissions is approximately a predefined target error rate.

A packet will typically contain more bits than can be transmitted in a single slot interval. Hence, a packet transmission will span a variable number of slot intervals. The number of slot intervals is random, depending on the realizations of the channel processes, and chosen parameters.

In simulations with multiple wireless channels, the fading processes of the channels are independent and identically distributed. At the completion of a packet transmission, the base station selects the user with the best channel quality, of those which have at least one packet queued at the base station pending transmission. The scheduling decision is only made once the previous packet transmission has completed, either successfully or unsuccessfully.

All experiments had a maximum data rate of $C_{\max} = 1.44$ Mbps. Because the channel rate, μ_c, was not constant, the value used in calculating $p(q)$ from (1) was the mean value over the preceding 100 packets.

4.2. Mice and elephants

It is well known that the majority of TCP flows are short ("mice"), while a significant amount of the congestion is caused by long flows ("elephants"). The throughput of mice is dominated not by their flow control scheme, but by the RTT they experience, because of the way TCP slow start works. This RTT is in turn determined by the queueing induced by the elephants.

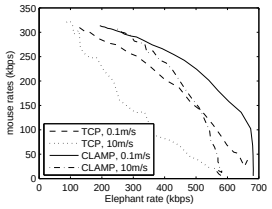
By keeping the queue short but non-zero, CLAMP allows both mice and elephants to achieve high throughput. This experiment consists of 26 flows sharing a single wireless link. One is an elephant, transmitting for the entire experiment. The remainder are mice, each transmitting only five packets of 500 bytes. They start at random times, uniformly distributed over a 30 s interval. There are eight mice with base RTTs (excluding queueing) of 15 ms, nine with base RTTs of 40 ms, and eight with RTTs of 300 ms. All these flows share a common wireless channel to the same wireless receiver.

By adjusting the parameter a in (1), a base station running CLAMP can control the tradeoff between the throughput achieved by mice and that achieved by elephants. A smaller value results in a smaller queue, and higher throughput for the mice, while a larger value results in the buffer emptying less often, increasing the throughput for the elephant. In TCP, the tradeoff can only be controlled by varying the buffer size at the base station. Figure 2 shows the effect of each of these mechanisms.

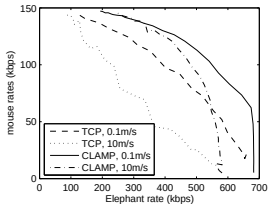
In Figures 2(a) and 2(b), it is assumed that the sender is using TCP Veno [15], which disregards wireless errors. This is modeled by setting the wireless packet error rate to zero. The elephant's base RTT is 80 ms. Figure 2(a) shows the throughput of a mouse with a very short 15 ms RTT, while Figure 2(b) is for the case of 40 ms. In both cases, CLAMP provides a significantly better tradeoff between the throughput of mice and elephants. When the mouse's RTT is higher, there is a very clear knee in the curve for CLAMP, making the selection of operating point straightforward. This occurs since the small amount of queueing required to ensure high utilisation causes a negligible reduction in the mouse's overall RTT. For TCP, no such knee exists, and it is impossible

to achieve high throughput for both mice and elephants.

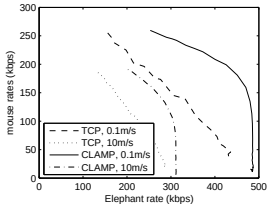
Most current TCP senders do not use Veno, and so CLAMP's performance was also investigated using Reno with a packet error rate of 10^{-3}, and the results are presented in Figures 2(c) and 2(d). In this case, the elephant's RTT was reduced to 40 ms, typical of the RTT between nearby cities. Due to the wireless errors, Reno is not able to maintain a large CWND, and CLAMP's effectiveness is reduced when CWND is smaller than AWND. Once again, CLAMP clearly provides a significantly better tradeoff of rates, and a clearer desired operating point than TCP does.



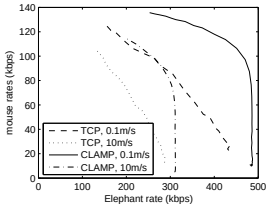
(a) Lossless, mouse RTT 15 ms



(b) Lossless, mouse RTT 40 ms



(c) Packet error rate 10^{-3}, mouse RTT 15 ms



(d) Packet error rate 10^{-3}, mouse RTT 40 ms

Figure 2: Throughput of mice as queueing is varied to vary elephant throughput.

4.3. Multiple receivers and multi-user diversity

In this section, we show that even the elephant flows can benefit from a CLAMP-controlled AP. In these experiments, we have multiple flows, each with a separate queue at the AP. The AP scheduler schedules the packet on the best link at any given time, and we therefore expect to see the benefit of multi-user diversity [16, 17], provided the queues can be kept nonempty. On completion of each successful packet transmission, the base station selects the next user to whom it should transmit to be the user with the highest SNR. If all queues are permanently full (so that the scheduler can pick the best of three channels), the peak channel throughput is slightly less than 1 Mbps, with the precise value depending on the packet loss rate considered. All links are bidirectional, with characteristics as shown in Table 1. Other simulation parameters are given in Table 2.

To demonstrate how CLAMP can be used to control

Table 1: Link configuration

Node 1	Node 2	Capacity	Delay	Pk loss rate
S_i	X	10 Mb/s	$d_i/2$	0
X	R_i	see text	1 msec	10^{-2}, 10^{-3}

Table 2: Simulation Parameters

Parameter	Value
TCP Packet Size	500 Bytes
CLAMP τ	5000 Bytes/s
CLAMP $\bar{\Delta}$	10000 Bytes
CLAMP b	1
CLAMP a	Variable
d_1	0.1345 s
d_2	0.041 s
d_3	0.2843 s
d_4	0.002525 s

the tradeoff between queueing delay and radio link utilisation, we ran several simulations based on the topology depicted in Figure 4. Experiments were run for a wide range of values for parameter a, giving a corresponding range of mean queue sizes.

The average time to download 1 MByte of data (after slow-start) are presented in Figure 3. With a packet loss rate of 0.001, these results indicate a factor of two improvement by using CLAMP when the mean queue size is 1 kBtye (giving about 30 ms delay).

One of the benefits of CLAMP is that the queues at the access point are less often empty, allowing maximum multi-user diversity gain [16, 17]. This is part of the reason for the throughput gains depicted in Figure 3. This might suggest that there is less gain when there is only one significant flow at the access point, a situation that might be quite common in many small wireless access networks, such as home LANs. However, when the propagation delay of the flow is large, CLAMP prevents buffer overflow and the resulting channel idleness. This effect is less significant when there is statistical multiplexing gain at the access point, from many flows sharing the same access point. To test this, we simulated flow 1 over the wireless channel, when the other flows are absent, and obtained a very similar throughput delay curve to that of Figure 3.

As an aside, note that to obtain the mean delay it is not sufficient to divide the mean queue size by the mean transmission rate. Because the rate varies, and there is positive correlation between having a slow channel and a large queue, this significantly underestimates the delay incurred. The above figures were obtained by explicit measurement of queueing delay and buffer occupancy.

4.4. Fairness

CLAMP's fairness is shown explicitly in Figure 5, which shows the throughput of each flow against the mean queueing delay (averaged over all flows). For low

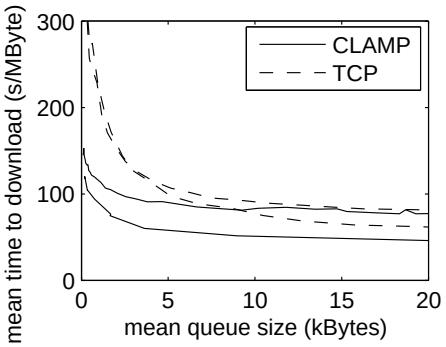


Figure 3: Time/Latency tradeoff for packet loss rates of 0.001 (lower curves) and 0.01 (upper curves). A queue size of 3500 bytes introduces a delay of approximately 85 ms.

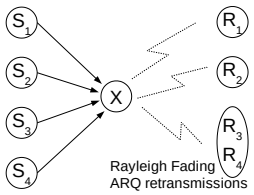
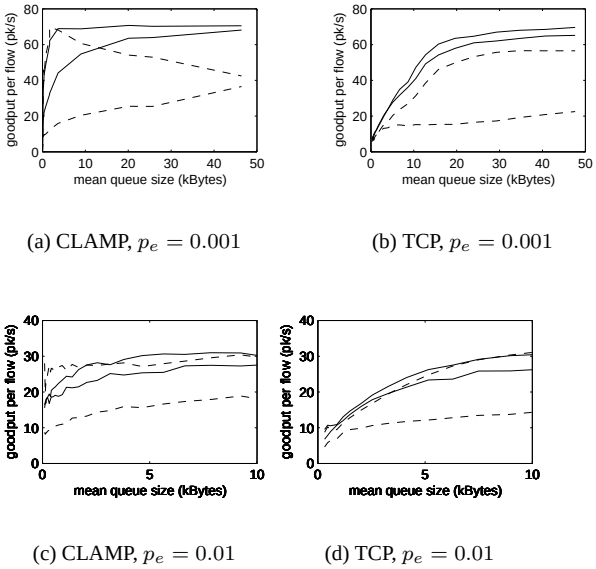


Figure 4: Simulation network topology

packet loss rates, (Figure 5(a)), flows 3 and 4 share their common queue fairly under CLAMP, despite a difference in round trip time of two orders of magnitude. Under TCP (Figure 5(b), these flows have very unequal throughputs, except when the queueing delay swamps the propagation delay giving them more equal round trip times (beyond the scale depicted).

When the packet loss rate is higher, CLAMP is no longer fair, but no less so than TCP alone (Figures 5(c) and 5(d)). CLAMP was designed for a reliable wireless link layer. Recent research has strongly advocated the design of very reliable wireless link layers [4–7,18]. Figures 5(a) and 5(c) indicate that, at a packet loss rate of 0.01, the inadequacy of TCP in dealing with loss dominates over CLAMP's efforts to provide fairness. This is the limit imposed by TCP halving its window on packet loss.

We remark that wireless packet loss is not an issue at all if wireless loss is detected at the sender, as occurs with TCP Veno. CLAMP will perform better with TCP Veno, than with TCP Reno, as the above figures attest. TCP must certainly react to packet loss due to congestion in the core network. Experiments in [10], show that CLAMP does not interfere with this aspect of flow control, and effectively hands over control to TCP Reno when a bottleneck occurs elsewhere in the network.



(a) CLAMP, $p_e = 0.001$ (b) TCP, $p_e = 0.001$

(c) CLAMP, $p_e = 0.01$ (d) TCP, $p_e = 0.01$

Figure 5: Throughput per flow for packet loss rates of $p_e = 0.001$ and $p_e = 0.01$. Dashed lines are for flows 3 and 4, which are on the same host.

5. Conclusions

Detailed packet level simulations of CLAMP running over test network topologies with multiple flows, and fading wireless channels, were performed. The results of the simulations indicate that total throughput, and individual file download times, can be improved whilst simultaneously decreasing latency for all the flows. The benefit of the decreased latency is most striking when considering CLAMP's impact on the throughput of short TCP flows, which are shown to benefit by up to 700%, with negligible impact on the throughput of the long TCP flows, in experiments over a wireless fading channel. This arises from CLAMP's ability to control the queue at the AP, to ensure that its average size remains no larger than is necessary to ensure full utilization of the wireless channel.

REFERENCES

[1] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Commun. Mag.*, vol. 38, pp. 54–64, Jan. 2000.

[2] D. Cygan and E. Offer, "Short linear incremental redundancy codes having optimal weight structure profile," *IEEE Trans. Inform. Theory*, vol. 37, pp. 192–195, Jan. 1991.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for

improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, pp. 756–769, 1997.

[4] H. M. Chaskar, T. Lakshman, and U. Madhow, "TCP over wireless with link level error control: Analysis and design methodology," *IEEE/ACM Trans. Networking*, vol. 7, pp. 605–615, Oct. 1999.

[5] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," in *Proc. IEEE Int. Conf. Netw. Protocls (ICNP)*, pp. 327–338, Mar. 2003.

[6] R. Ludwig, A. Konrad, A. Joseph, and R. Katz, "Optimizing the end-to-end performance of reliable flows over wireless links," *Kluwer/ACM Wireless Netw. J.*, vol. 8, pp. 289–299, Mar.-May 2002.

[7] M. Meyer, "TCP performance over GPRS," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1248–1252, Mar. 2003.

[8] H.-K. Shiu, Y.-H. Chang, T.-C. Hou, and C.-S. Wu, "Performance analysis of TCP over wireless link with dedicated buffers and link level error control," in *IEEE Int. Conf. Commun.*, pp. 3211– 3216, IEEE, 2001.

[9] M. Sagfors, R. Ludwig, M. Meyer, and J. Peisa, "Queue management for TCP traffic over 3G links," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1663–1668, Mar. 2003.

[10] L. L. Andrew, S. V. Hanly, and R. Mukhtar, "CLAMP: A system to enhance the performance of wireless access networks," in *Proc. IEEE Globecom*, 2003.

[11] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "Analysis of rate adjustment by managing inflows," in *Proc. 4th Asian Control Conference*, (Singapore), pp. 47–52, 2002.

[12] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt, "Flow aggregation for enhanced TCP over wide-area wireless," in *Proc. IEEE INFOCOM*, pp. 1754–1764, 2003.

[13] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "CLAMP: Maximizing the performance of TCP over low bandwidth variable rate access links," Technical report 2004-02-01, CUBIN, University of Melbourne, 2004. http://www.unimelb.edu.au/staff/lha/abstract/clamp_tr-2004-02-01.pdf.

[14] Information Sciences Institute University of Southern California, "RFC 793: Transmission control protocol," RFC 793, IETF, 1981.

[15] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Select. Areas Commun.*, vol. 21, pp. 216–228, Feb. 2003.

[16] R. Knopp and P. Humblet, "Information capacity and power control in single-cell multiuser communications," in *In Proc. IEEE International Conference on Communications*, vol. 1 of *ICC*, (Seattle), pp. 18–22, IEEE, IEEE, June 1995.

[17] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and A. Viterbi, "CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users," *IEEE Commun. Mag.*, vol. 38, pp. 70–77, July 2000.

[18] Y. Bai, A. Ogielski, and G. Wu, "Interactions of TCP and radio link ARQ protocol," in *IEEE Vehic. Technol. Conf.*, vol. 3, pp. 1710–1714, 1999.